AMENDMENTS TO THE CLAIMS

This listing of the claims will replace all prior versions, and listings, of claims in the application:

1. (Currently Amended) A method of generating an intermediate representation of a register-based program code~~,~~ that refers to a set of registers, the method comprising the computer implemented steps of:

generating a plurality of register objects each representing ~~abstract registers, a single~~ a respective one of said registers ~~object representing a respective abstract register; and~~ as referenced by the program code;

generating one or more expression objects each representing a respective operator or operand ~~different element~~ of said program code as that element arises in the program code~~,~~; and

forming a network of said register objects and ~~each~~ expression objects with each said expression object being referenced by one or more of said ~~a~~ register objects to which it relates either directly, or indirectly via references from other of said expression objects.

2. (Currently Amended) ~~A~~ The method according to claim 1, wherein said program code is expressed in terms of an instruction set of a subject processor.

3. (Currently Amended) ~~A~~ The method according to claim 2, wherein said register objects represent the set of ~~abstract~~ registers ~~corresponding to registers~~ of said subject processor.

4. (Currently Amended) ~~A~~ The method according to claim 1, further comprising the step of dividing ~~wherein each of~~ said program code into a plurality of ~~steps are performed sequentially for~~ basic blocks each ~~of said program code~~ having only one effective entry point instruction and one effective exit point instruction~~.~~, and performing said generating steps sequentially block-by-block with respect to said plurality of basic blocks.

5. (Currently Amended) ~~A~~ The method according to claim 1, wherein at least some said expression objects feed into more than one said register objects.

6. (Currently Amended) ~~A~~ The method according to claim 1, wherein said expression objects are not duplicated.

7. (Currently Amended) ~~A~~ The method according to claim 1, wherein a single said expression object is generated for a given operator or operand ~~element~~ of said program code, and each said expression object is referenced by all said register objects to which it relates.

8. (Currently Amended) ~~A~~ The method according to claim 1, further comprising the step of eliminating one or more ~~wherein if a~~ said register objects or a said expression objects ~~becomes~~ if they are identified as being redundant or unnecessary ~~it is eliminated~~.

9. (Currently Amended) ~~A~~ The method according to claim 8, further comprising the step of identifying ~~wherein~~ a redundant or unnecessary said register object or said expression object ~~is identified~~ by maintaining an ongoing count of references being made to that object as ~~a~~ the network of register and expression objects is constructed in said intermediate representation.

10. (Currently Amended) The method according to claim 9, wherein for each expression object a count is maintained of the number of references to that expression object from other expression objects or from register objects, the count associated with a particular expression object being adjusted each time a reference to that expression object is made or removed.

11. (Currently Amended) ~~A~~ The method according to claim 10, wherein an expression object and all references from that expression object are eliminated when said count for that expression object is zero.

12. (Currently Amended) The method of claim 1, further comprising the step of translating the program code written for execution by a processor of a first type so that the program code may be executed by a processor of a second type, using the generated intermediate representation.

13. (Currently Amended) ~~A~~ The method of claim 12, wherein said translating step is performed dynamically as the program code is run.

14. (Currently Amended) ~~A~~ The method of claim 1, further comprising the step of optimising the program code by optimising the generated intermediate representation.

15. (Currently Amended) ~~A~~ The method of claim 14, wherein said optimising step is used to optimise the program code written for execution by a processor of a first type so that the program code may be executed more efficiently by that processor.

16. (Currently Amended) A method for generating an intermediate representation of a register-based program code written for running on a programmable machine having a set of registers, said method comprising:

(i) generating a plurality of register objects in the intermediate representation, each said register object representing a respective one of the set of registers as referenced ~~for holding variable values to be generated~~ by the program code; and

(ii) generating a plurality of expression objects in the intermediate representation, said expression objects representing fixed values and/or relationships between said fixed values and said ~~variable values~~ registers according to said program code;

wherein said register objects and said expression objects are ~~being~~ organised into a branched tree-like network having all register objects at the lowest basic root or tree-trunk level of the network with ~~no~~ none of said register objects feeding into any other of said register objects.

17. (Currently Amended) A system for generating an intermediate representation of a register-based program code which refers to a set of registers, comprising:

means for generating a plurality of register objects each representing a respective one of said ~~abstract~~ registers~~, a single register object representing a respective abstract register; and~~ as referenced by the program code.

means for generating one or more expression objects each representing a ~~different element~~ respective operator or operand of said program code as that element arises in the program code~~,~~; and

forming a network of said register objects and ~~each~~ expression objects with each said expression object being referenced by one or more of said ~~a~~ register objects to which it relates either directly, or indirectly via references from other of said expression objects.

18. (Currently Amended) A system for generating an intermediate representation of a register-based program code written for running on a programmable machine having a set of registers, the system comprising:

means for generating a plurality of register objects, each said register object representing a respective one of the set of registers as referenced ~~for holding variable values to be generated~~ by the program code; and

means for generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said ~~variable values~~ said registers according to said program code;

wherein said register objects and said expression objects are organised into a branched tree-like network having all of said register objects at the lowest basic root or tree-trunk level of the network with ~~no~~ none of said register objects feeding into any other of said register objects.